**Eagleslist**

**Project Management Plan**

**CEN 3031, Fall, 2015**

**Modification history:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **Who** | **Comment** |
| **v1.0** | **October 5, 2015** | **Mostofa Ansary, Hermilo Calderon, Michael MacCallum,**  **Andrew Owen,**  **John Zachariah** | **Initial update to our Project Management Plan for the Eagleslist Application.** |
| **...** |  |  |  |

**Team Name:** 5Bit Studio

**Team Members:**

* Mostofa Ansary – msansary6258@eagle.fgcu.edu
* Hermilo Calderon – hcalder@eagle.fgcu.edu
* Michael MacCallum – msmaccallum5438@eagle.fgcu.edu
* Andrew Owen – adowen5642@eagle.fgcu.edu
* John Zachariah – jfzachariah3593@eagle.fgcu.edu

5-bit.github.io

**Contents of this Document**

(1) Project Overview

(2) Applicable Standards

* Coding Conventions
* Code Organization
* KISS (Keep it Simple Stupid)
* DRY (Don't Repeat Yourself)

(3) Project Team Organization

(4) Timeline

(5) Software Life Cycle Process

(6) Tools and Computing Environment

(7) Configuration Management

(8) Quality Assurance

* Test Coverage
* Code Reviews

**SECTION 1: Project Overview**

Eagleslist is a textbook focused trading program for FGCU students that will allow users to make accounts, verify their student email addresses and post used textbooks and other goods that they’re looking to buy or sell. We plan on adding special features for textbooks, like the ability to lookup books by CRN, or ISBN. Professors will also be able to verify their faculty email addresses and have the ability to supply the ISBNs for books that they will require for specific classes. Students will be allowed to comment and/or review course books to rate them by relevance/necessity/etc. We will not be handling money, all transactions will be determined by the students who buy and sell products on our service.

**SECTION 2: Applicable Standards**

Coding Conventions

In terms of coding conventions, our team will strive to follow existing conventions for things like variable names, indentation and general code formatting. For Go, we will be following the conventions listed in golang.org's0 Effective Go1, and for C#, we will be following Microsoft's C# Coding Conventions2 programming guide.

Code Organization

All files in our project, and methods within classes should be grouped logically by their functionality. This means that we will always attempt to have methods that are related to each other in groups, and related files stored together in folders. We will attempt to follow these rules, even if doing so requires refactoring the project because we believe this will make the project easier to maintain long term.

KISS (Keep it Simple Stupid)

When writing code for our project, all team members will try to write code with readability in mind and try to refrain from "cool tricks" that may sacrifice clarity, even if they produce slight performance gains or less total code.

DRY (Don't Repeat Yourself)

Repeated code will not be tolerated. If ever in a situation where the same code is required more than once, our team members will refactor the duplicated code into a method, function or similar so that the code only exists once within the project.

0 https://www.golang.org

1 https://golang.org/doc/effective\_go.html

2 https://msdn.microsoft.com/en-us/library/ff926074.aspx

3 https://www.github.com

4 https://help.github.com/articles/using-pull-requests/

**SECTION 3: Project Team Organization**

Team members:

* Andrew Owen - Head Manager (Oversees project development, team lead for web service)
* Michael MacCallum - Co-Manager (Track meetings and keeps website up to date, GUI team lead)
* Mostofa Ansary - Front-end programmer (Implement part of GUI, tester)
* Hermilo Calderon - Front-end programmer (Implement part of GUI, testing manager)
* John Zachariah - Back-end programmer (Implement parts of web service, responsible for idea)

Since our team members vary in experience, jobs and artifacts will be assigned to team members based on expertise (indicated above) and availability. Communication is a combination of face to face communication after classes and remote communication team communication, primarily Google hangouts. All team members will be given some responsibility for testing their code. Hermilo will be responsible for ensuring that each member has tested their code to work well.

**SECTION 4: Timeline**

|  |  |  |
| --- | --- | --- |
| Artifact | Due Dates | Team member(s) responsible |
| Meeting Minutes | Every Thursday at Holmes Hall Room 403/ Student Lounge at 2:30 - 6:30 PM | Every team member |
| Project Plan (document) | September 21, 2015 | Every team member |
| SRS (document) | September 30, 2015 | Every team member |
| Prototype | October 15, 2015 | Every team member |
| Design | October 22, 2015 | Every team member |
| Test Results | October 30, 2015 | Every team member |
| User’s Manual (document) | November 30, 2015 | Every team member |
| Source, Executable, Build Instructions | November 30, 2015 | Every team member |
| Presentation 1 | October 29, 2015 | Every team member |
| Presentation 2 | December 10, 2015 | Every team member |

**SECTION 5: Software Life Cycle Process**

We have decided to use the Increment and Iterate Lifecycle Model. The team members have already surveyed the students around the campus and collected a list of features that students wanted to have in our Eagleslist application. We are going to design, debug and test all of the features that the students would like to have. We closely chose to model real-world software production.

**SECTION 6: Tools and Computing Environment**

* Windows 7+ to run.
* Communication with a web service, and will therefore require an internet connection to operate.
* A system will at least 1GB of RAM, and at least a 1 GHz processor.
* C# and .NET 4.5
* Google Books API
* Web Service
* PostgreSQL
* Go
* Ubuntu 14.04
* Digital Ocean
* Journal Website
* HTML
* CSS
* Javascript
* Jekyll
* Liquid

**SECTION 7: Configuration Management**

We are hosting our code on GitHub, so server administration will be outsourced. Michael MacCallum will be responsible for integrating changes to the front end client’s repository and Andrew Owen will be responsible for integrating changes into the back-end web service’s repository. Both ends will use Pull requests to propose changes for review, and each lead will decide on a deployment process for their end. Deployment of changes will be on a rotating assignment to different team members.

**SECTION 8: Quality Assurance**

Test Coverage

We will strive to have high test coverage in our entire project, both in the WinForms application, and for our web service. All unit tests should make sure that every method is capable of behaving in an expected way for both correct and incorrect input. Updates to the code base that include fixes for any previously observed bugs should always include at least one unit test demonstrating the problematic behavior, and showing that the code no longer malfunctions.

Code Reviews

Even though we will all have write access to our collective GitHub3 repositories, and could just commit changes to their master branches directly, all changes to the master branches should be made in the form of pull requests4. Doing so will allow other members of the group to review differences in the code before they are merged into the master branch. We believe that working like this will help us catch potential problems with updated code before it causes problems with the project.

Template created by G. Walton (GWalton@mail.ucf.edu) on Aug 30, 1999 and last updated Aug 15, 2000, modified by A. Koufakou Aug. 2014

This page last modified by 5Bit Studio (jfzachariah3593@eagle.fgcu.edu) on 10/5/15